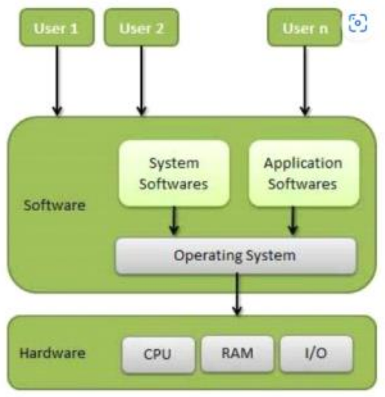
OPERATING SYSTEM (OS) NOTES

19 February 2023 17:07

1. WHAT IS AN OPERATING SYSTEM (OS) ?

An operating system is a program that acts as an intermediary between the user and the computer hardware. The purpose of an OS is to provide a convenient environment in which user can execute program in a convenient and efficient manner. An operating system is a software which performs all the basic tasks like file management, memory management, process management, handling input and output, and controlling peripheral devices such as disk drives and printers. For Example- Android OS, Mac OS (Macintosh), Linux, Windows, Unix, etc are different kinds of Operating systems.



Memory Management

Memory management refers to management of Primary Memory or Main Memory. Main memory is a large array of words or bytes where each word or byte has its own address. Main memory provides a fast storage that can be accessed directly by the CPU. For a program to be executed, it must in the main memory. An Operating System does the following activities for memory management –

- Keeps tracks of primary memory, i.e., what part of it are in use by whom, what part are not in use.
- In multiprogramming, the OS decides which process will get memory when and how much.
- Allocates the memory when a process requests it to do so.
- De-allocates the memory when a process no longer needs it or has been terminated.

Processor Management

In multiprogramming environment, the OS decides which process gets the processor when and for how much time. This function is called **process scheduling**. An Operating System does the following activities for processor management –

- Keeps tracks of processor and status of process. The program responsible for this task is known as **traffic controller**.
- Allocates the processor (CPU) to a process.
- De-allocates processor when a process is no longer required.

Device Management

An Operating System manages device communication via their respective drivers. It does the following activities for device management –

- Keeps tracks of all devices. Program responsible for this task is known as the **I/O controller**.
- Decides which process gets the device when and for how much time.
- Allocates the device in the efficient way.
- De-allocates devices.

File Management

A file system is normally organized into directories for easy navigation and usage. These directories may contain files and other directions.
An Operating System does the following activities for file management –

- Keeps track of information, location, uses, status etc. The collective facilities are often known as **file system**.
- Decides who gets the resources.
- Allocates the resources.
- De-allocates the resources.

2. WHY USE AN OPERATING SYSTEM (OS) ?

As a User

As a front user, the operating system helps a lot in interaction with the hardware. It helps as-

- **User interface –** OS provides a user interface with the help of which users can interact with the machines and instruct them a. Without User Interface it becomes hard for the users to interact with the machine as each user has to know about the machine language to give inputs to the machine.
- **Graphical Interface –** OS provides a graphical screen such as the icons. And with the help of the cursor user can interact with the computer to operate it.
- **Multitasking –** OS also allows the users for multitasking. This means the users can run more than one application simultaneously. And this is achieved by the operating system using the concept of threads and context switching. In Context switching processes are swapped in and swapped out from the CPU and processes are swapped in and out millions of times in a second to achieve multithreading. **Example of Multitasking –** Chatting while watching videos, Browsing while listening to music, Downloading while browsing, etc.

As a Programmer

- In **C Programming** Language we often used **printf** and **scanf** etc functions for input-output operation without knowing how it is internally processed to display on the console or retrieve from the console. This is managed by the operating system internally. We as a programmer need not define how to display on the console.
- Suppose if we develop an application then the amount of memory needed and processing power needed etc things we need not tell how to do it. Instead, we ask the operating system to do this for us.

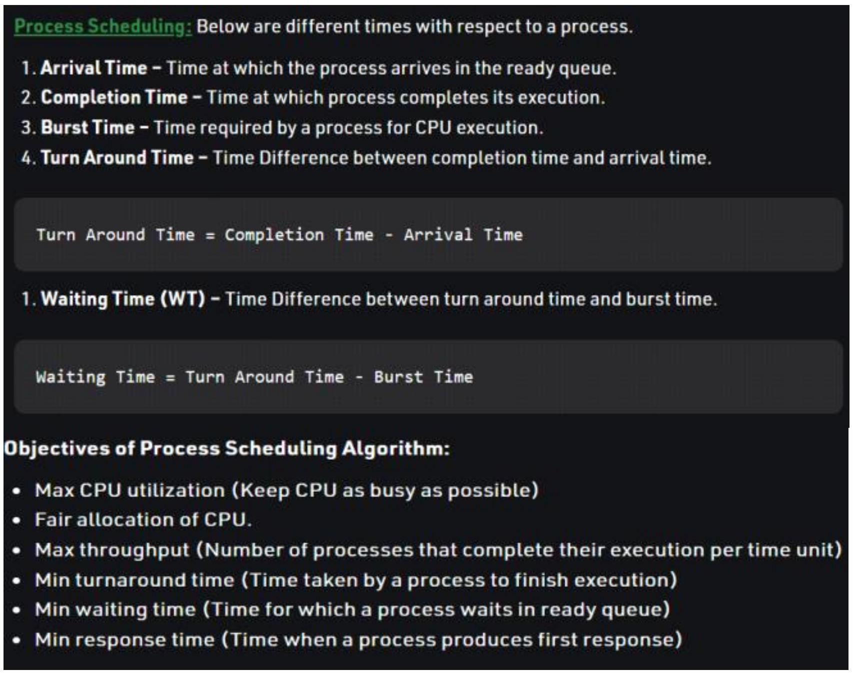
3. TYPES OF OPERATING SYSTEM (OS) ?

- **Batch Processing OS** – A set of similar jobs are stored in the main memory for execution. A job gets assigned to the CPU, only when the execution of the previous job completes.
- **Multiprogramming OS** – The main memory consists of jobs waiting for CPU time. The OS selects one of the processes and assigns it to the CPU. Whenever the executing process needs to wait for any other operation like (I/O), the OS selects another process from the job queue and assigns it to the CPU . This way the CPU is never kept idle and the user gets the flavour of getting multiple tasks done at once.

- **Multitasking OS** – Multitasking OS combines the benefits of multiprogramming OS and CPU scheduling to perform quick switches between jobs. The s witch is so quick that the user can interact with each programs as it runs.
- **Time-Sharing OS –** In this, each program has been assigned a particular time slot by the OS. And the programs are allowed to be executed only within these time slots. This is to achieve the concept of multiprogramming.
- **Distributed OS –** In this, Operating System is installed in the context of managing multiple hardware. Hardware includes the (CPU, Main Memory, Storage). Each hardware resource is individually performed processing and distributed OS helps to allocate processes to be executed on the hardware.
- **Real-time OS –** It is a software component that rapidly switches between tasks, giving the impression that multiple programs are being executed at the same time on a single processing core.

3. WHAT IS A PROCESS ?

A process is a program under execution. Each process is represented by a Process Control Block.



Process Scheduling: Below are different times with respect to a process.

1. **Arrival Time –** Time at which the process arrives in the ready queue.
2. **Completion Time –** Time at which process completes its execution.
3. **Burst Time –** Time required by a process for CPU execution.
4. **Turn Around Time –** Time Difference between completion time and arrival time.

```
Turn Around Time = Completion Time - Arrival Time
```

1. **Waiting Time (WT) –** Time Difference between turn around time and burst time.

```
Waiting Time = Turn Around Time - Burst Time
```

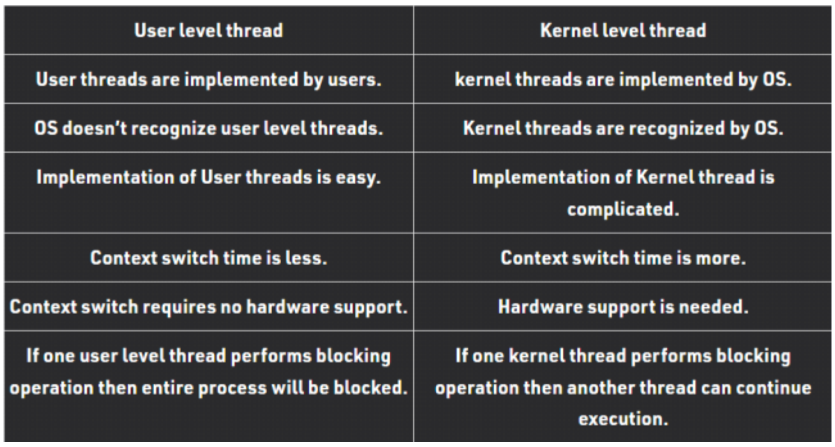
Objectives of Process Scheduling Algorithm:

- Max CPU utilization (Keep CPU as busy as possible)
- Fair allocation of CPU.
- Max throughput (Number of processes that complete their execution per time unit)
- Min turnaround time (Time taken by a process to finish execution)
- Min waiting time (Time for which a process waits in ready queue)
- Min response time (Time when a process produces first response)

3. WHAT IS A THREAD ?

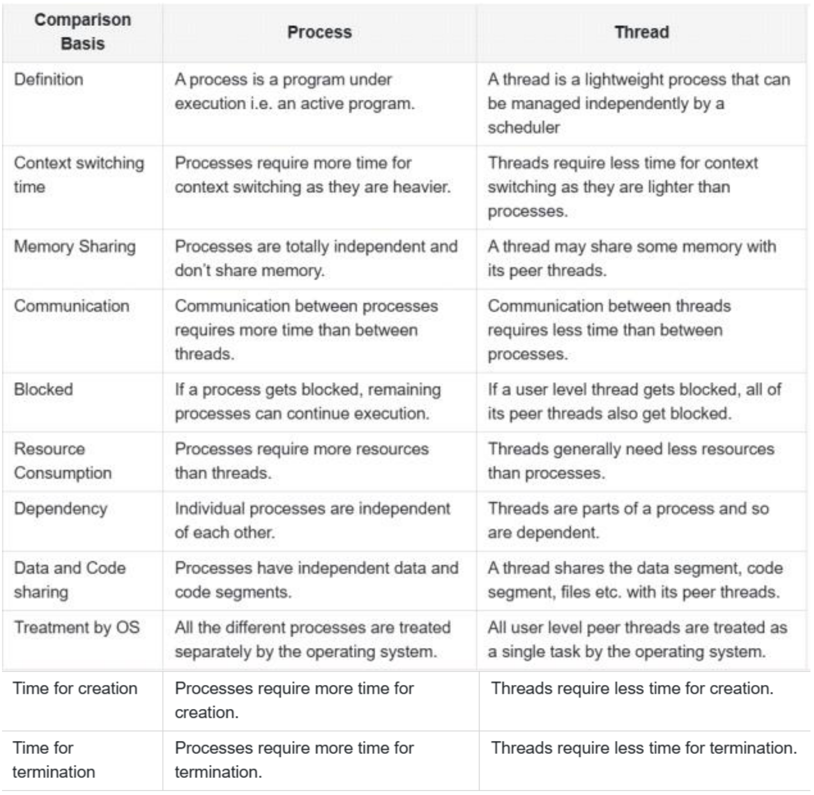
A thread is a lightweight process and forms the basic unit of CPU utilization. A process can perform more than one task at the same time by including multiple threads. A thread has its own program counter, register set and stack. There are 2 types of threads:

. User threads
. Kernel threads



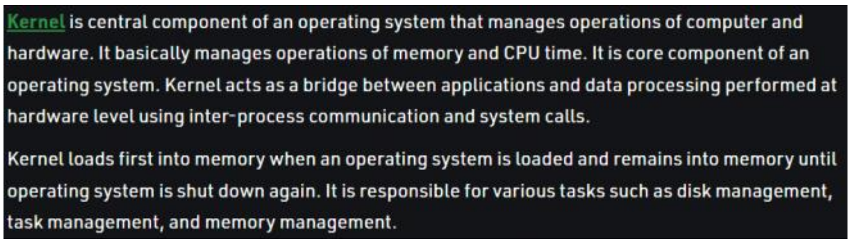
User level thread	Kernel level thread
User threads are implemented by users.	kernel threads are implemented by OS.
OS doesn't recognize user level threads.	Kernel threads are recognized by OS.
Implementation of User threads is easy.	Implementation of Kernel thread is complicated.
Context switch time is less.	Context switch time is more.
Context switch requires no hardware support.	Hardware support is needed.
If one user level thread performs blocking operation then entire process will be blocked.	If one kernel thread performs blocking operation then another thread can continue execution.

4. DIFFERENCE BETWEEN PROCESS AND THREAD ?



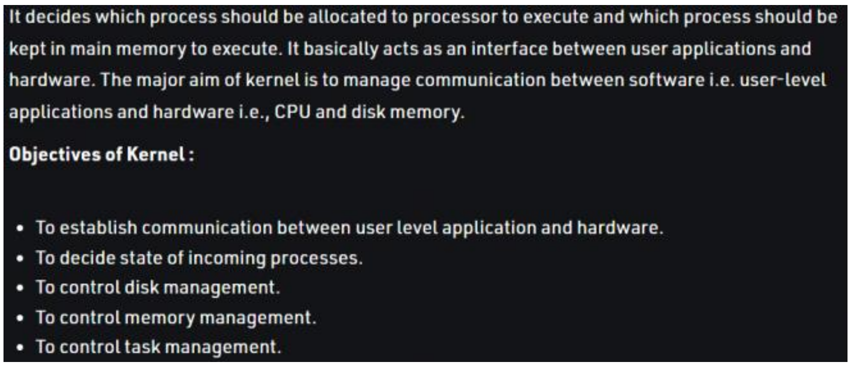
Comparison Basis	Process	Thread
Definition	A process is a program under execution i.e. an active program.	A thread is a lightweight process that can be managed independently by a scheduler
Context switching time	Processes require more time for context switching as they are heavier.	Threads require less time for context switching as they are lighter than processes.
Memory Sharing	Processes are totally independent and don't share memory.	A thread may share some memory with its peer threads.
Communication	Communication between processes requires more time than between threads.	Communication between threads requires less time than between processes.
Blocked	If a process gets blocked, remaining processes can continue execution.	If a user level thread gets blocked, all of its peer threads also get blocked.
Resource Consumption	Processes require more resources than threads.	Threads generally need less resources than processes.
Dependency	Individual processes are independent of each other.	Threads are parts of a process and so are dependent.
Data and Code sharing	Processes have independent data and code segments.	A thread shares the data segment, code segment, files etc. with its peer threads.
Treatment by OS	All the different processes are treated separately by the operating system.	All user level peer threads are treated as a single task by the operating system.
Time for creation	Processes require more time for creation.	Threads require less time for creation.
Time for termination	Processes require more time for termination.	Threads require less time for termination.

5. WHAT IS A KERNEL ?



Kernel is central component of an operating system that manages operations of computer and hardware. It basically manages operations of memory and CPU time. It is core component of an operating system. Kernel acts as a bridge between applications and data processing performed at hardware level using inter-process communication and system calls.

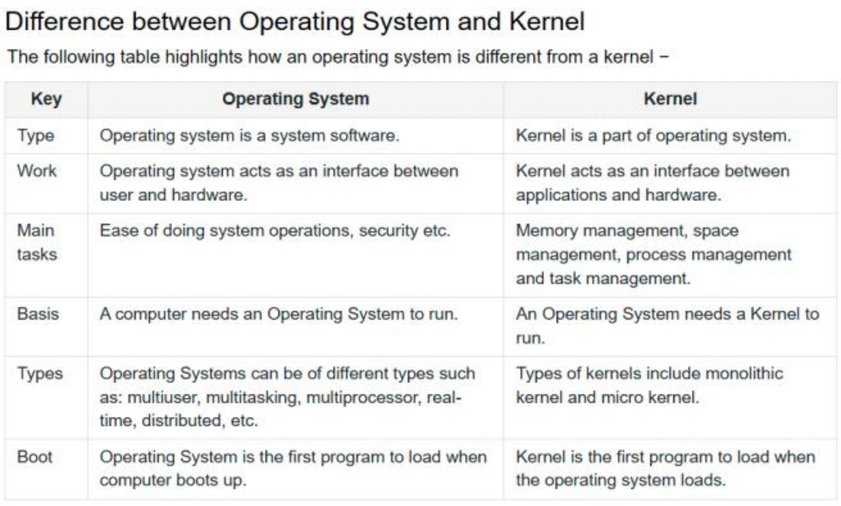
Kernel loads first into memory when an operating system is loaded and remains into memory until operating system is shut down again. It is responsible for various tasks such as disk management, task management, and memory management.



It decides which process should be allocated to processor to execute and which process should be kept in main memory to execute. It basically acts as an interface between user applications and hardware. The major aim of kernel is to manage communication between software i.e. user-level applications and hardware i.e., CPU and disk memory.

Objectives of Kernel :

- To establish communication between user level application and hardware.
- To decide state of incoming processes.
- To control disk management.
- To control memory management.
- To control task management.



Difference between Operating System and Kernel

The following table highlights how an operating system is different from a kernel –

Key	Operating System	Kernel
Type	Operating system is a system software.	Kernel is a part of operating system.
Work	Operating system acts as an interface between user and hardware.	Kernel acts as an interface between applications and hardware.
Main tasks	Ease of doing system operations, security etc.	Memory management, space management, process management and task management.
Basis	A computer needs an Operating System to run.	An Operating System needs a Kernel to run.
Types	Operating Systems can be of different types such as: multiuser, multitasking, multiprocessor, real-time, distributed, etc.	Types of kernels include monolithic kernel and micro kernel.
Boot	Operating System is the first program to load when computer boots up.	Kernel is the first program to load when the operating system loads.

6. DIFFERENCE BETWEEN MONOLITHIC & MICROLITHIC KERNEL ?

What is Microkernel

Microkernel is a type of kernel that allows customization of the operating system. It runs on privileged mode and provides low-level address space management and Inter Process Communication (IPC). Moreover, OS services such as file system, virtual memory manager, and CPU scheduler are on top of the microkernel. Each service has its own address space to make them secure. Besides, the applications also have their own address spaces. Therefore, there is protection among applications, OS Services and kernel.

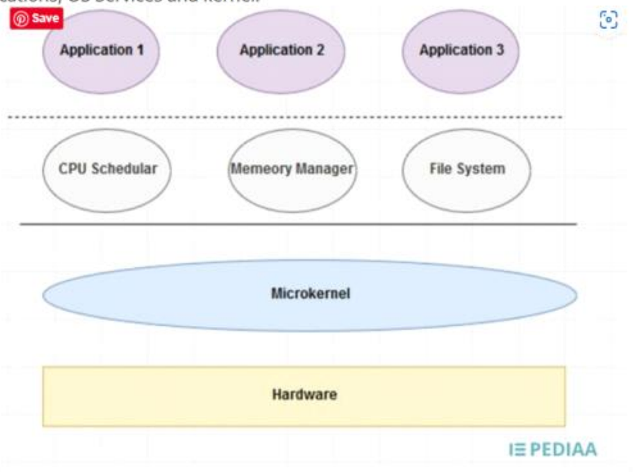
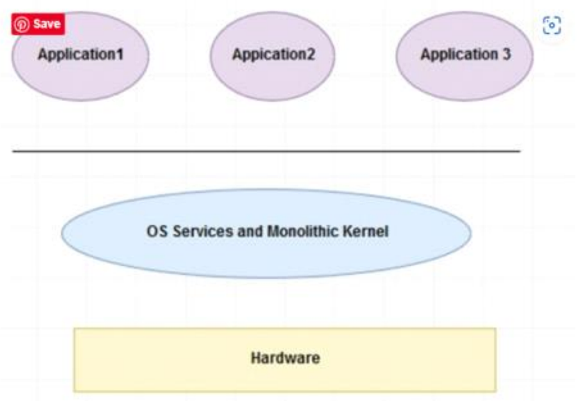


Figure 1: Microkernel Architecture

When the application requests the OS services for a service, the OS services communicate with each other to provide the required service to the application. Here, the Inter Process Communication (IPC) helps to establish this communication. Overall, microkernel-based OS provides a great level of extensibility. It is also possible to customize the services of the operating system depending on the requirements of the application.

What is Monolithic Kernel

In monolithic kernel based systems, each application has its own address space. Therefore, each application is secure. Also, the kernel contains all the OS services. Therefore, the applications can request services from the kernel. Some OS services are file system, CPU Scheduler, network access, memory manager etc. However, the OS is in a separate address space. Therefore, it is secure from the normal applications and malfunctioned applications.



If an application requires a service, the hardware address space of the application switches to the hardware address space of the operating system to execute it.

OS Services

A notable difference between microkernel and monolithic kernel is that, in a microkernel-based system, the OS services and kernel are separated. But, in a monolithic kernel system, the kernel contains the OS services.

Speed

Speed is also a major difference between microkernel and monolithic kernel. A microkernel system is slow while monolithic kernel system is fast.

Failures

In a microkernel-based system, failure in one component will not affect the other components. However, in a monolithic kernel based system, failure in one component will affect the entire system. Hence, this is also an important difference between microkernel and monolithic kernel.

Customization

Another difference between microkernel and monolithic kernel is that it is easier to add new functionalities to the microkernel; therefore, it is more customizable. However, it is difficult to add new functionalities to the monolithic kernel; therefore, it is not customizable.

Magnitude

Moreover, the microkernel is smaller in size while the monolithic kernel is larger in size.

7. WHAT IS CONTEXT SWITCHING ?

Context switching in the operating system signifies the storage of the state or context of a specific process in such a way that it can be reloaded whenever it is needed to resume execution right from where it was left.
In technical terms, this is a process that involves multitasking with the operating system that allows sharing only one CPU by several processes and by switching between them.

8. WHAT IS LONG-TERM SCHEDULER AND SHORT-TERM SCHEDULER ?

Long-term Schedulers are the job schedulers that select processes from the job queue and load them into memory for execution.
Short-term Schedulers are the CPU schedulers that select a process from the ready queue and allocate the CPU to one of them.

9. WHAT IS PRE-EMPTIVE & NON-PRE-EMPTIVE SCHEDULING

In Pre-emptive scheduling, CPU can be pre-empted from a running process. The highest priority process should always be the process that is currently utilized.
In Non-Pre-emptive scheduling, the CPU can't be pre-empted from a running process till process completes or leaves CPU by itself.

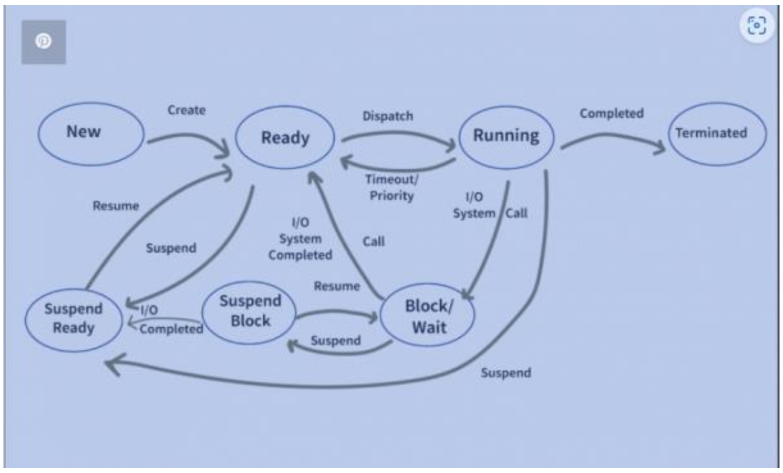
10. WHAT IS DISPATCHER ?

A dispatcher gives control of CPU to the process selected by the short-term scheduler. It involves switching context, switching to user modes and jumping to the location in the user program to restart that program.

11. WHAT IS SPOOLING ?

Refer to this Link :- Spooling in Operating System - javatpoint

12. WHAT ARE THE VARIOUS STATES OF PROCESS ?



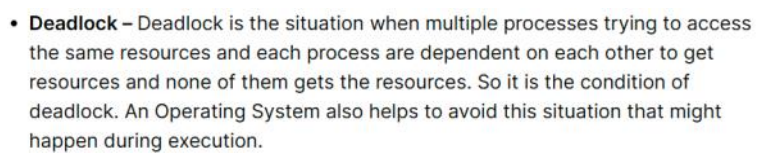
13. WHAT ARE THE VARIOUS SCHEDULING ALOGORITHMS ?

Refer to this Link :- Operating System Scheduling algorithms (tutorialspoint.com)

14. WHAT IS STARVATION AND AGING IN OS ?

Starvation and Aging in Operating Systems - GeeksforGeeks

15. WHAT IS DEADLOCK ?



- **Deadlock –** Deadlock is the situation when multiple processes trying to access the same resources and each process are dependent on each other to get resources and none of them gets the resources. So it is the condition of deadlock. An Operating System also helps to avoid this situation that might happen during execution.

Necessary conditions for DeadLock :-

1. Mutual Exclusion (where atleast one resource is non-shareable)
2. No pre-emption (where the resources can't be pre-empted)
3. Hold & wait (where a process holds one resource and waits for other resource)
4. Circular wait (where processes are waiting for each other in a circular loop fashion)

Methods for handling DeadLock :-

1. Deadlock Ignorance (Ostrich method)
2. Deadlock prevention (above 4 conditions)
3. Deadlock avoidance (Bankers Algo) Banker's Algorithm in Operating System (OS) - javatpoint
4. Deadlock detection & Recovery)

16. WHAT IS THRASHING ?

Thrashing in OS (Operating System) | What is Thrash? - Javatpoint

17. WHAT IS PROCESS SYNCHRONIZATION ?

Introduction of Process Synchronization - GeeksforGeeks

18. WHAT IS OVERLAY ?

Overlays in Memory Management - GeeksforGeeks

19. WHAT IS VIRTUAL MEMORY ?

What is Virtual Memory in OS (Operating System)? - javatpoint

20. WHAT IS THE DIFFERENCE BETWEEN PAGING AND SEGMENTATION ?

Difference between Paging and Segmentation (tutorialspoint.com)